

Dyne.org GSoC 2010

Dyne Hackers

2010-03-11

Contents

1 Dyne.org Application :GSoC2010:	1
1.1 About dyne.org	1
1.1.1 Community	2
1.1.2 History	2
1.1.3 Why we apply	3
1.1.4 Projects Licensing	3
1.2 Contact	3
1.3 Ideas	4
1.3.1 FreeJ :freej:	4
1.3.2 PSYC :psyc:	5
1.3.3 Frei0r :frei0r:	6
1.3.4 Cyberdeck :cyberdeck:	6
1.3.5 Infopoint :infopoint:	7
1.3.6 MuSE :muse:	8
1.4 Mentors	8
1.4.1 Criteria for Mentor Selection	8
1.4.2 Acracia :infopoint:	9
1.4.3 Asbesto :muse:	9
1.4.4 Caedes :freej:	9
1.4.5 Crash :psyc:	9
1.4.6 Hellekin :psyc:	9
1.4.7 Jaromil :freej:muse:frei0r:	9
1.4.8 lynX :psyc:	9
1.4.9 Newmark :cyberdeck:	9
1.4.10 Xant :freej:muse:	10
1.5 Process	10
1.5.1 How to Deal With Disappearing Students?	10
1.5.2 How to Deal With Disappearing Mentors?	10
1.5.3 How to Encourage Students to Interact With the Community Before, During and After GSoC	10
1.5.4 How to Ensure Selected Students Will Stick to The Project After GSOC	11

1 Dyne.org Application :GSoC2010:

This *ongoing* document (also available in pure text and PDF) is the central point for information about the Google Summer of Code 2010 at dyne.org.

file:images/aliens/2010soclogo.jpg

The GSoC is a program where Google pays students stipends to work over the summer on free software projects. Each student works with one or more mentors (in our case, dyne.org's hackers) to complete a development task.

If we are selected as a mentoring organization, 2010 will be the first participation of dyne.org in the program, a very important step for our informal yet solid network of developers which exists since 2000.

update: unfortunately, Dyne.org was not selected to be a mentoring organization this year. But we are determined to continue support for the projects listed here. So, if you're interested, stay tuned and pop up in the channel to get involved!

1.1 About dyne.org



Dyne.org is a **non-profit** effort lead by a **grassroot** committee of **hackers** dedicated to development of **free and open source software** for the **freedom of expression**.

One dyne is the force required to cause a mass of one gram to accelerate at a rate of one centimeter per second squared in the absence of other force-producing effects.

A dyne is 1.000.000 times a Newton.

The dyne measure has been established by Heraclitus, a greek philosopher born at Ephesus around 540 b.C., whom once also said that **much learning does not teach understanding**.

1.1.1 Community

A young and displaced sisterhood of hackers, alchemists, radio amateurs, mathematicians and nomads.

The Freaknet is an on-site medialab, museum and poetry hacklab in the Mediterranean island of Sicily, surviving since 1994 the hostile environment of South Italian criminal administration and cultural repression.

The Hackmeeting is since 1998 the annual gathering of computer and reality hackers in Italy and Spain, an auto-organized TAZ inspired by people and projects at CCC, [[<http://2600.org/>]²⁶⁰⁰], GNU and EFF, which was recently awarded the AEC prixars 2009 honorary mention for digital communities.

Servus.at supported the birth of this digital community since the very beginning, hosting it in the digital space, offering solidarity and support for our on-line operations.

Since 2005 we are registered as a non-profit Foundation in the Netherlands (well able to fill an IRS form W8-BEN), thanks to the kind hospitality of the NIMK, formerly known as Montevideo / Time Based Arts: a national institute that trusted our work and started adopting our Free Software R&D in the artistic field.

Our digital community is born from these on-site communities and even more on-line, it gathered a wider international participation interacting on **software development** and **network cultures**.

1.1.2 History

Dyne.org appeared online in 2000 when the HasciiCam software was published: an invention widely appreciated for its artistic value and for making possible to broadcast live video using old hardware from a slow network connection.

Once upon a time dyne.org homepage consisted of its skin-like coloured background, the upper menu recited “korova, muse, ascii, proximity, theorema, timezones, conspire” while a transforming Moebius band decorated the center of the page, tagged with a quote from **Gertrude Stein**: *a rose is a rose is a rose is a rose is a rose*



Inspired by a mix of **software** and **poetry**, a growing network of developers released to the public software made to insure **freedom of expression**, configuring dyne.org as a free software atelier, a portal to **Digital Creation** and **Media Art**.

dyne.org software is redistributed by: * Free Software Foundation (USA) * Nederlands Instituut voor Media Kunst (NL) * Ircam, Centre Pompidou (FR) * Vienna Univ. of Technology (AT) * Providence Univ. Taichung (TW) * Technische Univ. Ilmenau (DE) * Netherlands Unix User Group (NL) * Instituto de Computação Uni de Campinas (BR) * Heraklion University Crete (GR) * Ibiblio public library and many others.

Ranging from radio makers, humanitarian organizations, video artists, medical researchers, media activists and educators, a large amount of people employed and redistributed dyne.org software worldwide, free of charge, echoing to the freedom spirit of this autonomous initiative.

Openness, knowledge sharing and freedom of creation have been the philosophical principles guiding the evolution of our network, hosting creations that have been conceptualised not for a profit, but for their role within society.

1.1.3 Why we apply

Because it's time to do so.

We didn't jumped on the train of Google SOC at its very beginning for two main reasons:

- * we wanted to see what it was really going to be
- * we weren't stable enough to be proper mentors

Today, after being busy for years around “consolidation plans”, we are sure we can be worth the effort and accomplish good results. We are also sure Google SoC’s is an innovative project, definitely aligned with our support for Free Software as well as our social engagement and academic background.

Many of us already work with students in schools or occasional workshops, running various events and most importantly getting people involved: we’d love to give students a chance to learn more by doing, and earning some money might really help some brilliant developers to emerge from the economically depressed southern economies connected to our network.

At last, we really need to keep our code tight and can really use some help on that: our network at large is asking for it and we even have the big responsibility of maintaining one of the few 100% Free GNU/Linux distribution recommended by the Free Software Foundation and more software used by free speech activists around the world.

1.1.4 Projects Licensing

All the code written by developers in our community, as well as the code adopted to run our on-line and on-site infrastructure, is Free Software along the principles and recommendations set by the GNU project.

When releasing code we prefer to adopt the GNU General Public License version 3 and, in case developers requires it, other licenses approved by the FSF.

1.2 Contact

- GSoC2010 URL: <http://dyne.org/gsoc2010>
- Mailing-List: gsoc-uhuh-lists.dyne.org
- Instant messaging over PSYC (multiprotocol and much tendentious)
 - IRC: <irc://hinezumi.im/dyne>
 - Web: <https://hinezumi.im:8443/?channels=dyne>
 - Jabber: xmpp:*dyne@hinezumi.im
 - Telnet: <telnet://hinezumi.im> (pick a name, then /go dyne)
- Student Application Template: <http://dyne.org/gsoc2010-student-application>
- GSoC Organization Administrators: Jaromil and Hellekin

1.3 Ideas

1.3.1 FreeJ :freej:

Summary	Vision Mixer
Required Skills	C++, Javascript, Python, OSX GUI design
Mentors	Jaromil, Xant, Caedes
URL	http://freej.dyne.org
REPO	git://code.dyne.org/freej.git

FreeJ is a **vision mixer**: an instrument for realtime video manipulation used in the fields of dance teather, veejaying, medical visualisation and TV.

- **TODO**

- * Write a graphical user interface using QT or GTK, possibly using the flowcanvas widget or similar approaches to modular synths
- * Write a web browser plugin for Mozilla Firefox and/or Google Chrome
- * Adopt v8-juice to add a javascript parser using Google's V8
- * Design a graphical user interface for VeeJaying and streaming on Apple OSX, also using the XCode Cocoa graphic designer. The whole application is already ported for native usage of Quicktime components, but its interface lacks a good design from a user perspective.
- * Organise existing javascript examples and write more documentation and tutorials, for example create an interactive tutorial in javascript running in FreeJ

- What we need to do

We need to interact with **multiple layers** of video, filtered by **effect chains** and then **mixed together**.

We write scripts to control video mixes with keyboards, midi signals, OSC messages, wiimotes, video mouse and joysticks; manipulating images, movies, live cameras, particle generators, text scrollers, flash animations and more.

We us the resulting video mix on **multiple and remote screens**, **encode it into a movie** and **stream it live** to the internet.

We **control the vision mixer locally or remotely**, also from multiple places at the same time; most functionalities are **designed ad-hoc using javascript**.

- How FreeJ does it

FreeJ is a commandline application on GNU/Linux, a graphical application on Apple/OSX, a C++ library offering an API for a multimedia framework that relies on different native functions on the operating systems it is ported and, at last, bindings to languages like Python and Ruby (using Swig, more can be implemented as needed).

The code is fairly documented and usable in C++, with full **bindings to python**, while parsing scripts is done using Spidermonkey, the Mozilla interpreter (there are plans to use also Google V8 in the future).

FreeJ has started being developed on a dual-core CPU already in 2001 and has grown with emphasis on multi-threading to run efficiently on modern multi-core computers.

- Internals

FreeJ is written with efficiency in mind, benefits of a realtime object oriented and multi-threaded architecture where layers and controllers all run independently, to take advantage of multiple CPUs and clustered systems.

The language employed in development is C/C++ respecting POSIX compliance and avoiding the computational bloat of some 'advanced' C++ functions, which makes it highly portable. The FreeJ Debian package for instance is also distributed in binaries for ARM and MIPS processors.

Its C++ programming API is fairly understandable, here you'll find a brief introduction to it.

Here is an overview of packages and quality assurance provided by Debian GNU/Linux. Packages are available since years also for Ubuntu, Fedora and Arch distributions, while we maintain our own build server to agevolate deployment tests.

1.3.2 PSYC :psyc:

Summary	Protocol for SYnchronous Conferencing
Required Skills	LPC or Pike, C/C++, ECMAscript and XUL, perl, python, ruby
Mentors	lynX, Hellekin, Crash
URL	http://about.psyce.eu/
REPO	git://git.psyced.org/git/psyced

PSYC is a protocol for a messaging infrastructure for text-based conferencing, but also enabling transparent binary data. It has learned from protocols such as IRC and XMPP and found an approach that should indeed scale globally, by generalizing the multicast concept beyond chatrooms to presence awareness, news- and friendcasting and peers at decentralized privacy-driven social networking (using trust metrics), telephony and audio/video conferencing. Chatrooms are programmable, which enables for applications such as event notification.

There are many ways to contribute to PSYC. Depending on your programming skills and interests, you'd rather tackle the interserver protocol specification and testing, its server implementation (psyced uses LPC, there are plans for writing server code in other languages), the common library of core functions in C or some client development (e.g. jsPSYC and Psyczilla).

At Dyne, our confidence in the capabilities of PSYC made us adopt it for our daily communications, and we firmly intend to shed some well-deserved light on that project.

Whether you're interested in network protocols or global scalability, social networking or instant messaging, push and real-time applications, or decentralized networking architecture and software design, you'll find some aspect of PSYC to get you started.

If you ever got into MUDs, you'll find your way in the psyced server implementation in LPC. Core functionality for all platforms is being added to a C library. Pike, Perl and Python hackers will enjoy playing with the respective PSYC libraries in that languages. There's one in Ruby on the way. On the client front, Psyczilla brings PSYC to the desktop and browser using the Mozilla XUL framework. . .

But beware, PSYC has been maturing over a decade and your reading skills will be put to test! The PSYC team is mostly based in Europe.

• TODO

- High level web interface for the decentralized social network functions
- Multicast cryptography implementation (Crypto chatroom)
- Refactor psyced to fully support the current spec and expectations of PSYC
- Protocol interoperation test suite
- Distributed multicast filesystem

- Multicast editing / multicast extension to Google Wave
- More ideas for software projects

1.3.3 Frei0r :frei0r:

Summary	Free video plugins
Required Skills	Pixel juggling, C or C++ basic knowledge
Mentors	Jaromil
URL	http://frei0r.dyne.org
REPO	git://code.dyne.org/frei0r.git

Frei0r is a minimalistic plugin API for video effects.

The main emphasis of its design is on simplicity: an API that rounds up the most common video effects into simple filters, sources and mixers, to be controlled by parameters.

- **TODO**

This project is particularly accessible to people beginning to program, will give a good introduction to packaging and portability standards while engaging the student with some simple C/C++ coding to write new plugins or port existing ones to its API.

* Port plugins from the EffecTV collection * Port plugins from the Demo-Effects collection * Port plugins from Pete Warden's collection

While this idea mostly consists in simple porting of code, we believe this task is still very important for the correct preservation of this code, historically dealing with realtime video manipulation, which could be made available to more free software applications.

- Outreach

Frei0r video plugins are in use among many free software video applications as Gephex, LiVES, FreeJ, MøB, VeeJay, Open Movie Editor, pdvjtools, DVEdit, MLT and KDeLive among the others.

Eventually the frei0r API can be wrapped by higher level APIs expanding its functionalities (for instance, gstreamer does it).

1.3.4 Cyberdeck :cyberdeck:

Summary	Cyberdeck Collectible Card Game Clients
Required Skills	SDL / C++ or Javascript / HTML5
Mentors	Newmark, Xant

Cyberdeck is a collectible card game that simulates a cyberspace war among computers networks and their operators.

- **TODO**

Points below are listed as open possibilities for students to get involved with the project and choose their area of interest and skills, not all of them have to be achieved, just one can be chosen:

* create a webbased client for Cyberdeck that can handle also all the other gccg supported games (HTML5 or Ajax based).

- * implement rules into gccg server (in order to know if a move is legal or not).
- * create a specific C++/SDL (multiplatform or at least osx/linux) client for Cyberdeck that can control the validity of a move, wait for the opponent input when needed and so on.

- Introduction

In some way Cyberdeck is a mixed breed because is not only a card game but it has also some features of a strategy board game. The scenario is a mix between realistic computer technology of today and futuristic features inspired by the cyberpunk literature, mainly by the novels of William Gibson (above all the first trilogy: Neuromancer, Count Zero and Monnalisa Overdrive). (see <http://cyberdeck.dyne.org>)

Actually Cyberdeck can be played in real life, printing the cards provided, or on GCCG (see <http://gccg.sourceforge.com>) that is a is a multiplayer multiplatform implementation of a card game engine. GCCG provide server and client, with the limit that the client and the server don't know anything about the rules of any game. Just provide a table, a market and a deck management.

1.3.5 Infopoint :infopoint:

Summary	non-hierarchical organization framework
Required Skills	python django user interface xmpp
Mentors	Tatiana de la O
URL	http://abbadingo.cryptodrinks.net/?p=292
REPO	git://code.dyne.org/infopoint.git

The infopoint is a system to organize a social space in a non-hierarchical way. Why do we want to make a new scheduler? There are several applications on the open source business side of things to handle schedules/resource allocation. But all of them are business-minded.

We would like to develop a tool to help others to work in loose organizations, where people come to collaborate once a week but is not there every day, for example.

The infopoint is coded in Django, a web framework done in python.

- Planned Features

- Resource allocation (or “I need the beamer next friday”)
- free/busy information per user
- iCal support
- bluetooth: leave a picture for the social center
- report generation
 - * activity reports
 - * calendar
- meeting-helpers
- integration with jabber / psyc
- integration with mailing lists

That project is mentored by a woman whose previous *minions* praised and reclaimed. You'll be treated fine under her lashŴŴŴ.

1.3.6 MuSE :muse:

Summary	Internet radio streamer
Required Skills	C++, audio DSP, GTK2
Mentors	Jaromil, Xant, Asbesto
URL	http://muse.dyne.org
REPO	git://code.dyne.org/muse.git

MuSE provides the free software community with a user friendly but powerful tool for network audio streaming, making life easier for independent free speech online radios.

- **TODO**

MuSE was the first free Internet audio streaming software to provide a graphical interface for on-line radio makers, back in 2000: it started with a TCL/TK interface, then had an FLTK one, then a GTK1 and now finally a GTK2 - plus at last has been ported on OSX (Carbon/Cocoa GUI). Its usage is well documented and widespread.

As you can imagine, its code is quite old and now quite in the need of a good revamp: cleanup the pipes and tight up the buffers, simplify some code parts and substitute some old libraries with newer ones (especially integrating use of FFMpeg and GStreamer).

The object oriented approach of its code makes it easy to approach and to substitute various implementations, so this task is mostly about studying API of existing libraries and implement them as C++ classes inside MuSE.

- **Description**

MuSE can mix up to 6 encoded audio bitstreams (from files or network, mp3 or ogg) plus a soundcard input signal, the resulting stream can be played locally on the sound card and/or encoded at different bitrates, recorded to harddisk and/or streamed to the net. When sent to a server, the resulting audio can be listened thru the net by a vast number of players available on different operating systems.

1.4 Mentors

1.4.1 Criteria for Mentor Selection

We selected the mentors primarily because of their direct involvement with the ideas proposed and their experience with the tasks at hand.

Besides the scope of GSoC we keep in contact and do our best to develop the projects and adopt them into our specific areas of activity.

1.4.2 Acracia :infopoint:

Tatiana de la O aka Acracia is nomadic developer, a Django fairy and a software artist developer of horizontal communication platforms; she goes by the motto “if I can’t hack it, it’s not my application.”

1.4.3 Asbesto :muse:

Gabriele Zaverio, aka Asbesto Molesto, can hardly be described in words. He spends all his time recycling hardware and making ancient computers work, like a crazy hunchback banging a PDP-11, builds laptops made in wood and ravel around assembler code on VAX-VMS.

He is a decision maker one can be happy to find in the middle of a thermonuclear attack.

1.4.4 Caedes :freej:

Pablo Martines, aka Caedes, is a game developer and crystal architect proficient in Python, OpenGL and most of the rest; he can also make APIs talk any language, pronouncing some magic SWIG formula and lighting up a funny candle.

1.4.5 Crash :psyc:

Manuel Cangemi aka crash is a Hinezumi/Freaknet hacker and technology visionary, he maintains the hinezumi PSYC server and hangs around the Via Crucis on the Etna Volcano.

1.4.6 Hellekin :psyc:

Hellekin O. Wolf is a Dyne hacker, PSYC and Hackerspaces cat herder, insatiable learner and convinced generalist. He prefers spending Winters hibernating IFOC and Summers traveling, (hitch-)hiking and breathing fresh air in good company. And a laptop.

1.4.7 Jaromil :freej:muse:frei0r:

Jaromil is a rasta coder (and pastafarian) who loves designing OO architectures and developing in C++ and ZSHaolin “for the digital revolution”. He is the prototype of a digital artisan who likes to challenge big tasks and lose sleep to free software, hiding himself in the downtown street shuffle of Am*dam or traveling around like a techno-beduin. Nevertheless he is 24/7 online.

1.4.8 lynX :psyc:

Carlo von Loesch, aka lynX, is the project leader of the PSYC protocol and the psyced server - and the one who invented the `/me` command for the IRC protocol.

1.4.9 Newmark :cyberdeck:

Pierluigi Maori aka Robert J. Newmark is a Perl hacker, project manager and expert gamer who enjoys interacting with people way more than hackers do on average.

1.4.10 Xant :freej:muse:

Andrea Guzzo aka Xantar Majere is a Unix software engineer with significant experiences in cross-platform software development and embedded solutions. He is most proficient in Perl, C++ and Obj-C, mostly developing on OSX and BSD platforms. Despite his rock-solid technical background he doesn't mind to interact with GUI designers and, sometimes, to think human.

1.5 Process

1.5.1 How to Deal With Disappearing Students?

Most of the ideas proposed are part of larger development frameworks or existing applications: students will add a contribution to larger activity pools, hopefully finding it interesting to experience collective development and interaction with our (often humorous) team.

In case students will disappear, we will document in detail their activity and acknowledge the partial efforts they have made. There will be no impact on the projects we'll propose, it's just an advantage for all of us (users included) if we stick together.

However our students never really disappear: we mark them with fire.

"I've found that coat hangers and a blow torch make great brands. They are easily malleable and will bend into wonderful designs that your kids will love. Grappa is a great disinfectant too."
– Matt Joyce, NYCResistor

1.5.2 How to Deal With Disappearing Mentors?

Our passionate Mentors won't defect: they're dedicated to the advancement of the project they chose.

Most of the ideas proposed are related to long running projects. For years, we've been building access to a clear infrastructure, consolidated practice, readable code and available documentation.

We assign 3 mentors to bigger projects, so that students won't be left alone.

In the unlikely event of mentors disappearing, we can count on our larger network to take care of the students.

1.5.3 How to Encourage Students to Interact With the Community Before, During and After GSoC

We have been working with students before and had welcomed involvement of people on a voluntary basis through the years. Thanks to various contacts within existing educational institutions around the Planet we will announce our joint initiative with Google to a large public.

Students who show up early on the list and the channel will get better chances to get enrolled. As requisites are matched and we get to know each other, the human factor kicks in and it's a matter of personality and will to stay in touch. We promote constant availability online and frequent interaction between developers.

With GIT we are able to track progress and provide a good degree of analysis on our activities.

The screen command provides a way for a mentor to pair-program remotely with a student, while conversing via PSYC (or IRC or XMPP, at the student's convenience).

We also use Emacs Org-mode to organize TODOs, calendars and documentation. Although not a requirement, it's a winning combo for tough players and really enhances our communication.

1.5.4 How to Ensure Selected Students Will Stick to The Project After GSOC

If some passion about the projects and their enthusiastic communities sparks in the students participating, it will be quite natural to keep in touch on-line and networking around new future possibilities. In fact, within dyne.org, we share many contacts and project calls that often create spontaneous collaborations among its members.

Recently, we are running coding sprints in places where we can access generous hosting and existing facilities: IRL meetings help to involve us and familiarise with each other spirits.

In addition to this, our close collaboration with the NIMK in the Netherlands will offer future possibilities for internships and stages, as well workshops that can be run within its facilities.

1.6 Timeline

Here are the next steps:

February 8		Program announced. Life is good.
March 8	~12 noon PST / 19:00 UTC	Mentoring organizations can begin submitting applications to Google.
March 12	4 PM PDT / 23:00 UTC	Mentoring organization application deadline.
March 13-17		Google program administrators review organization applications.
March 18	~12 noon PDT / 19:00 UTC	List of accepted mentoring organizations published on the Google site.
March 18-29		Would-be student participants discuss application ideas with mentors.
March 29	~12 noon PDT / 19:00 UTC	Student application period opens.
April 9	12 noon PDT / 19:00 UTC	Student application deadline.

The global timeline is available on GSoC site.

In case our application is accepted, we will publish the detailed calendar of our activities and meetings here.